Synergy Graphs for Configuring Robot Team Members

Somchaya Liemhetcharat
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
som@ri.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
veloso@cs.cmu.edu

ABSTRACT

Robots are becoming increasingly modular in their design, allowing different configurations of hardware and software, e.g., different wheels, sensors, and algorithms. We are interested in forming a multi-robot team by configuring each robot (i.e., selecting the different modules) to best fit a task. This general problem is applicable to many domains, such as manufacturing in high-mix low-volume scenarios. In this paper, we formally define the Synergy Graph for Configurable Robots (SGraCR) model, where each robot module is modeled as a vertex in a graph, and we define how to compute the synergy of modules within a single robot, as well as between robots, using the structure of the graph. We define the synergy of a multi-robot team comprised of such configurable robots, and contribute a team formation algorithm that searches a SGraCR to approximate the optimal team. In addition, we contribute a learning algorithm that learns a SGraCR from a small set of training data containing the performance of teams. We evaluate our SGraCR model and algorithm in extensive experiments, both in simulation and with real robots, and compare with competing algorithms.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

Capability; synergy; team formation; heterogeneous; modular; robots

1. INTRODUCTION

Multi-robot teams are commonly considered for performing complex tasks, and research has focused on the problem of task allocation and team formation, where the goal is to select the best subset of robots to perform a task. As robots have become more advanced and modular in their design, it is now feasible to consider what modules to include in the robots of a team. For example, when composing a multi-robot team, a robot that has a LIDAR is considered against a robot that is physically identical, except that it does not have the LIDAR but has a camera instead. We are interested in such a problem domain, where robots are composed of modules, and the goal is to configure each robot (i.e., select which modules to use) in the multi-robot team.

This robot configuration problem is general and applicable to many robot domains. For example, in manufacturing with high-mix low-volume scenarios, the manufacturing job floor has many robotic modules (such as drilling stations, milling stations, robotic arms, and mobile platforms that transport items). Each configuration forms a robot with certain capabilities, e.g., a mobile platform with an arm can lift and transport items. Given a new task, the goal is to select the appropriate modules that is feasible to complete the task in the shortest time. In addition, since new tasks may arrive while the selected modules are performing the task, there is also a need to keep the opportunity cost of using the modules below a certain threshold. Another example of the robot configuration problem is in mapping, where the robot modules are LIDARs, cameras, and motors. The interactions among modules are initially unknown: e.g., while faster motors are generally beneficial, it depends on other modules: a robot with cameras might have motion blur moving too quickly, decreasing performance. The goal would be forming a team that completes the mapping quickly, but stays below a maximum dollar cost of the modules.

Prior research in team formation and task allocation, that we elaborate in the related work section, focuses on robots with pre-defined modules and capabilities. We are interested in modeling the capabilities of the robotic modules, how well different modules work when placed into a single robot, and the interactions of multiple modular robots in a team.

We formally define the Synergy Graph for Configurable Robots (SGraCR) model, where each module is a vertex in a connected weighted graph, and the distances between vertices are related to how compatible they are. We define intra-robot synergy, that is the performance of modules within a robot, and inter-robot synergy, that is the performance of modules across robots. Using intra and inter synergy, we define the synergy of a multi-robot team composed of such configurable robots. We recently introduced the Synergy Graph model, where each agent is a vertex in a graph [5]. SGraCR models each module separately, and is more expressive with a lower number of vertices.

We contribute a team formation algorithm, that selects modules and composes them into robots of a multi-robot team and approximates the optimal team for the task. We contribute a learning algorithm, that learns a SGraCR using

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

a small set of training examples, by learning both the structure of the SGraCR structure and modules' capabilities.

We extensively evaluate our model and algorithms in a series of experiments. We first show that our learning and team formation algorithm performs well when learning data derived from a hidden SGraCR model. Next, we use our SGraCR model to learn data from simulated robots in a manufacturing scenario, and show that it outperforms competing algorithms. Finally, we show that our model and algorithms find effective teams in a real robot experiment.

2. RELATED WORK

Multi-robot task allocation (MRTA) is well-studied and market-based techniques are frequently used, where robots form bids on tasks, and tasks are awarded to the lowest bidder [4]. Robots can be modeled as lists of services or resources, and tasks are lists of required services/resources [8]. The main difference between services and resources is that each robot can only provide one service at a time, while all resources can be devoted to a task. Robots can also be modeled as schemas, where a schema is an operation with predefined inputs and outputs [9]. IQ-ASyMTRe forms coalitions of robots to solve multiple tasks by searching through executable robot schemas that complete a task [9]. While modeling robots as services, resources and schemas is similar to our approach of modules, a key difference in our work is that we do not assume that robots are predefined with their modules. Instead, we are interested in modular, configurable robots where module selection is done in order to configure effective robots for the team, and we compare our model and algorithms with IQ-ASyMTRe.

In agent-based task allocation, agents are similarly modeled as lists of resources, and social network graphs have been considered to model possible multi-agent teams [3]. We are interested in using task-based graphs for robot team formation, where edges indicate task-based relationships among robots instead of social ones. We recently introduced the Synergy Graph model, where agents are modeled as vertices in a connected unweighted graph, and a larger distance between agents in the graph indicates lower compatibility in a team [5]. In this work, we model each robot module as a separate vertex, and have edges with two weights to differentiate between compatibility of modules within a robot, and across different robots. Such a representation allows for a larger space of robot types to be modeled with a lower number of vertices, i.e., the number of vertices increases linearly compared to exponentially in the Synergy Graph model. We also compare our SGraCR model with the Synergy Graph.

Coalition formation involves the partitioning of a set of agents into disjoint subsets in order to maximize a value function. Typically, each coalition (i.e., subset of agents) is given an independent value, and the value of a partition is the sum of values of coalitions [7]. Recently, externalities in coalitions have been considered, where the value of a coalition depends on how other agents outside the coalition are grouped, e.g., with positive and negative externalities [6], or with mixed externalities [1]. We are interested in forming a single multi-robot team, and not the partition of robots into multiple teams. Also, we model the performance of a team based on which robots are configured to be in the team, and not due to external factors.

In ad hoc domains, the capabilities of robots and their performance as a team is initially unknown. Research in the ad hoc domain has focused on a single robot, e.g., how an ad hoc robot adjusts its behavior in a pursuit-evasion scenario [2]. We are interested in learning the capabilities of ad hoc robots, and do so by learning how well different robot modules work together in a multi-robot team.

3. DEFINING THE PROBLEM

In this section, we formally define the problem and give an overview of our solution. While we use a motivating manufacturing scenario to aid in the description of the problem, it is general and applicable to many other robotic domains.

3.1 Motivating Scenario

High-mix low-volume manufacturing is an emerging trend, where manufacturing plants have to manufacture a large variety of products, each with a low volume. This is in stark contrast to the typical manufacturing line that produces a large volume of a single product. In order to handle high-mix low-volume orders, manufacturing floors have to be reconfigurable and cater to each order. Each manufacturing station is viewed as a robot, and the entire manufacturing plant is a multi-robot system. Manufacturing stations include the typical drilling and milling, and also mobile robots that transport items. Each manufacturing robot is a configuration comprising one or more modules, e.g., a drilling robot comprises a drilling machine, and a mobile robot comprises motors and sensors. New types of robots can be configured, such as a robot that drills as it transports items.

Given a manufacturing task, the manufacturing plant has to select and configure manufacturing robots that will complete the task. The goal is to complete the task as quickly as possible, while limiting the cost of production. Robot modules have a fixed dollar cost that is borne by the manufacturing plant, and by assigning manufacturing modules to robots, the modules cannot be used for other orders and opportunity cost is lost. Hence, the multi-robot team that is formed is capped at a maximum level of cost decided by the plant, which is a function of the dollar cost and opportunity cost. In addition, the task has a pre-defined sequence of actions (e.g., drilling followed by milling followed by polishing), and the multi-robot team has to be capable of completing the task. In particular, random combinations of modules may not be able to perform the task (e.g., selecting a robot team that does not include any drilling machines).

3.2 Formal Problem Definition

Let $\mathcal{M} = M_1 \cup \ldots \cup M_N$ be the set of all modules, where each $M_n \in \mathcal{M}$ is a set of modules of type $n \in [1, \ldots, N]$. In the manufacturing scenario, M_1 would contain possible drilling modules, such that $M_1 = \{\text{drill}_0, \ldots, \text{drill}_3\}$ where the subscript refers to the number of drilling machines. M_2 would contain motors of different maximum speeds for a mobile robot, i.e., $M_2 = \{\text{none, slow, medium, fast}\}$.

Let $R = (m_1, \ldots, m_N)$ be a robot, where each $m_n \in M_n$, and let $\mathcal{R}$ be the set of all possible robots. Thus, each robot is a configuration/selection of modules of every type. In the manufacturing example, $R_0 = (\text{drill}_2, \text{none})$ is a stationary drilling robot with two drilling machines, while a mobile transportation robot is $R_1 = (\text{drill}_0, \text{medium})$. A robot that can drill and transport items is $R_2 = (\text{drill}_1, \text{slow})$.

Let $T : \mathcal{R} \rightarrow \mathbb{Z}_0^+$ be a team of robots, where $T(R)$ returns the number of robots that use the modules selected in R. Let $\mathcal{T}$ be the set of all possible teams. Let $F : \mathcal{T} \rightarrow \{0, 1\}$ be the feasibility function, where $F(T) = 1$ iff the team T is feasible to complete the task, and 0 otherwise. The

feasibility function F is domain-dependent and we assume that it is given as part of the problem definition.

Let $\text{cost}_{\mathcal{M}} : \mathcal{M} \rightarrow \mathbb{R}_0^+$ be the module cost function, and let $\text{cost}_{\mathcal{R}} : \mathcal{R} \rightarrow \mathbb{R}_0^+$ be the robot cost function, where $\text{cost}_{\mathcal{R}}(R) = \sum_{i=1}^{N} \text{cost}_{\mathcal{M}}(m_i)$. Similarly, let the cost function of robot teams be $\text{cost}_{\mathcal{T}} : \mathcal{T} \rightarrow \mathbb{R}_0^+$, where $\text{cost}_{\mathcal{T}}(T) = \sum_{R \in \mathcal{R}} T(R) \cdot \text{cost}_{\mathcal{R}}(R)$. As such, the cost of a robot is the sum of costs of its modules, and the cost of a robot team is the sum of costs of the robots in it. The module cost function is domain-dependent and part of the problem definition. In the manufacturing domain, the module cost function $\text{cost}_{\mathcal{M}}$ would be a function of the dollar cost of the module and the opportunity cost of using the module for the task.

Let $V(T)$ be the value attained by the robot team $T \in \mathcal{T}$. As we are interested in robots acting in a dynamic world, $V(T)$ is non-deterministic and multiple observations of $V(T)$ return different values. In the manufacturing example, $V(T)$ would be related to the time required for the team T to complete the manufacturing task and the cost of the team, e.g., a task that completes earlier with a lower cost team attains a higher value; the non-determinism would be related to potential failures in the manufacturing modules. The value function V is initially unknown and we seek to model it in order to form an effective multi-robot team.

The goal is to form a multi-robot team that attains the highest value subject to a maximum cost $c_{\max}$ while being feasible. However, since V is non-deterministic, we define the goal as forming the δ-optimal team T_δ^*, such that $P(V(T_\delta^*) \geq v) = \delta$, $\text{cost}_{\mathcal{T}}(T_\delta^*) \leq c_{\max}$, and $F(T_\delta^*) = 1$. For other $T \in \mathcal{T}$, $P(V(T) \geq v) \leq \delta$ if $\text{cost}_{\mathcal{T}}(T) \leq c_{\max}$ and $F(T) = 1$. Thus, the δ-optimal team is a feasible team that attains a value of at least v with probability δ and a cost threshold of $c_{\max}$, while any other feasible team does so with probability at most δ. If $\delta = 0.5$, then the δ-optimal team corresponds to the team that attains the highest mean value. We assume that the values $c_{\max}$ and δ are given and domain-specific, as are F and $\text{cost}_{\mathcal{M}}$ as mentioned earlier.

3.3 Overview of Approach and Contributions

We recently introduced the Synergy Graph model [5], that models and learns the synergy of multi-agent teams through observations of their performance. In this paper, we introduce a new model that handles configuring robots by treating every robot not as an atomic (indivisible) entity but as a combination of modules:

- We formally define the Synergy Graph for Configurable Robots (SGraCR) model, and how to compute the value of a multi-robot team;
- We contribute our team formation algorithm, that uses a SGraCR to approximate the δ-optimal team;
- We contribute our learning algorithm, that learns a SGraCR using only limited observations of possible robot teams;
- We demonstrate the efficacy of the SGraCR model and our algorithms in extensive experiments involving simulated and real robots.

4. MODELING MULTI-ROBOT SYNERGY

In the Synergy Graph model, agents are treated as atomic (indivisible) entities, and the agents' capabilities are modeled as Normally-distributed variables [5]. Fig. 1 shows an example of a Synergy Graph with 6 agents. Each agent is a vertex in a connected unweighted graph, and the distance between the agents in the graph is inversely related to their compatibility to work together in a team. The capabilities of agents are represented as Normally-distributed variables.

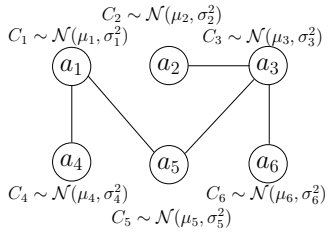


Figure 1: A Synergy Graph with 6 agents.

We introduce the Synergy Graph for Configurable Robots (SGraCR) model, that is specialized to model configurable modular robots performing multi-robot tasks. We first describe each component of our model, and give the formal definition at the end of this section.

4.1 Modular Representation of Robots

While treating agents as atomic entities allows an abstraction to capture humans and robots, we are interested in using the Synergy Graph for teams of robots. Robots are comprised of a configuration of various hardware and software modules. For example, a mobile exploring robot comprises hardware such as the motors, LIDAR, and other sensors, and software such as the SLAM algorithm.

Instead of treating agents/robots as indivisible entities (as done in the Synergy Graph model), we model each module as a separate vertex in a graph, which offers a large benefit in scalability. From the problem definition, there are N types of modules $M_1, \ldots, M_N$, and a robot is composed of one of each type of module. In our SGraCR model, there are $\sum_{n=1}^{N} |M_n|$ vertices; the Synergy Graph model would have $\prod_{n=1}^{N} |M_n|$ vertices. For example, suppose there are 3 different types of motors, 2 different LIDARs, 3 cameras, and 2 SLAM algorithms. By modeling each module as a vertex, our SGraCR would contain $3+2+3+2 = 10$ vertices. A Synergy Graph models each possible type of agent/robot separately with $3 \times 2 \times 3 \times 2 = 36$ vertices. Thus, the number of vertices increases linearly in SGraCR with the number of modules, while the Synergy Graph increases exponentially.

4.2 Synergy of Modules

We are interested in modeling the task-based performance of multi-robot teams, where each robot is composed of different modules. We build upon the Synergy Graph model, where the synergy of a multi-agent team is a combination of individual agent capabilities and their compatibility in the team [5]. Since the SGraCR models robots as composite modules, we need to differentiate between two types of synergy: intra-robot synergy and inter-robot synergy.

Intra-robot synergy models how the configuration of modules that compose a single robot affects how well the robot performs at the task. For example, a robot with faster motors performs the task quickly and attains a high performance. Comparatively, a robot with slightly slower motors but a more accurate vision system may be able to perform the task more accurately with even higher performance.

Inter-robot synergy models how different combinations of robots affect the overall task performance. Since the task requires multiple robots, different choices of robots in the

team will critically affect the task performance. For example, in a foraging task, a team consisting of a single robot with accurate vision and multiple fast-moving robots that retrieve the objects may perform better than a team with multiple robots with accurate vision but move more slowly.

Similar to the Synergy Graph model, we use Normally-distributed variables to represent the capability of each module. These variables represent the contribution of task performance from the module, subject to the synergy from intra and inter-robot relationships. We use Normally-distributed variables as performance is non-deterministic, and a random variable captures this variability; further, Normal distributions are commonly used and have many favorable characteristics that we use, such as having an equal mean and mode, and symmetric deviations. We use the distance between vertices in the graph to represent how well modules work together, as in the Synergy Graph model. However, there are two main differences in the SGraCR model. First, the SGraCR model uses weighted edges, while the Synergy Graph uses an unweighted graph; weighted edges allow for a larger representational space. Second, we distinguish between intra and inter-robot synergy by assigning two weights to each edge in the graph, which implies that the underlying structure of the graph (i.e., whether edges exist between vertices) are common between intra and inter-robot synergy, even though the weights may differ. Such a representation offers a more elegant structure (a single graph structure with multiple weights) compared to two independent graphs. Further, we believe that it is justified as there is a correlation between intra and inter-robot synergy — a module that works well for a robot will also benefit a multi-robot team.

In addition to the weighted graph structure, the SGraCR model has an additional edge associated with each vertex, that models the inter-robot synergy between the same module on different robots, e.g., the task performance of two robots that both have the same type of motors. An intra-robot weight in this case is not needed, since each individual robot cannot select multiple copies of the same module type.

4.3 The SGraCR Model

We have described the components of the SGraCR model above, and now we formally define it:

Definition 1. The **Synergy Graph for Configurable Robots** model is a tuple $\{G, C\}$, where:

- $G = (V, E)$ is a connected graph;
- Each $m \in \mathcal{M}$ is represented by a vertex $v_m \in V$;
- $e = (v_m, v_{m'}, w_{\text{intra}}, w_{\text{inter}}) \in E$ is an edge with two integer weights. w_{intra} and w_{inter} are the intra and inter-robot weights respectively (edge weights between modules on the same/different robot);
- $e_m = (v_m, v_m, w_{\text{inter}}) \in E$ is a self-looping edge with a single inter-robot weight;
- $C = \{C_1, \ldots, C_{|\mathcal{M}|}\}$ is a set of module capabilities, where $C_m \sim \mathcal{N}(\mu_m, \sigma_m^2)$ is the capability of a robotic module $m \in \mathcal{M}$.

We assume that module capabilities are independent, as dependencies and synergies among modules are captured by the SGraCR graph structure. Fig. 2 shows an example of a SGraCR with 6 vertices, where there are two types of modules with 3 options each. Compared to a Synergy Graph that models 6 agents with 6 vertices (Fig. 1), $3 \times 3 = 9$ different types of robots are represented with the SGraCR.

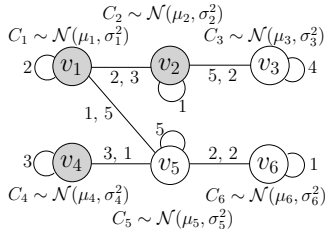


Figure 2: A SGraCR with 6 vertices, modeling two types of modules (shown in different shades). The edges with two weights indicate the intra and inter-robot weights respectively, and the self-looping edges have inter-robot weights.

Definition 2. The **intra-robot synergy** $\mathbb{S}_{\text{intra}}(R)$ of a robot $R = (m_1, \ldots, m_N)$ is:

$$\mathbb{S}_{\text{intra}}(R) = \sum_{m_i, m_j \in R} \phi(d_{\text{intra}}(v_{m_i}, v_{m_j}))(C_{m_i} + C_{m_j}) \quad (1)$$

where C_{m_i} and C_{m_j} are the capabilities of modules m_i and m_j respectively, $d_{\text{intra}}(v_{m_i}, v_{m_j})$ is the shortest distance between vertices v_{m_i} and v_{m_j} in the SGraCR using the intra-weights w_{intra} of the edges, and $\phi : \mathbb{Z}^+ \to \mathbb{R}^+$ is a compatibility function, which we describe below.

Definition 3. The **inter-robot synergy** $\mathbb{S}_{\text{inter}}(R, R')$ of two robots $R = (m_1, \ldots, m_N)$ and $R' = (m'_1, \ldots, m'_N)$ is:

$$\mathbb{S}_{\text{inter}}(R, R') = \sum_{m_i \in R, m'_j \in R'} \phi(d_{\text{inter}}(v_{m_i}, v_{m'_j}))(C_{m_i} + C_{m'_j}) \quad (2)$$

where C_{m_i} and $C_{m'_j}$ are the capabilities of modules m_i and m'_j respectively, $d_{\text{inter}}(v_{m_i}, v_{m'_j})$ is the shortest distance between vertices v_{m_i} and $v_{m'_j}$ in the SGraCR using the inter-robot weights w_{inter} of the edges. In particular, if $v_{m_i} = v_{m'_j}$ then the self-looping edge is used to determine d_{inter}.

The compatibility function $\phi : \mathbb{Z}^+ \to \mathbb{R}^+$ converts distances in the SGraCR graph to real numbers reflecting the compatibility among robot modules. ϕ is monotonically non-increasing, so larger distances correspond to equal or lower compatibility. In this way, modules that are more compatible are closer together in the SGraCR graph. We assume that ϕ is domain-specific and given. Some examples of ϕ are $\phi_{\text{fraction}}(d) = \frac{1}{d}$, and $\phi_{\text{decay}}(d) = \exp(-\frac{d \ln 2}{h})$.

Definition 4. The **synergy** $\mathbb{S}(T)$ of a multi-robot team $T : \mathcal{R} \to \mathbb{Z}_0^+$ is:

$$\mathbb{S}(T) = \frac{1}{|T|} \sum_{R \in \mathcal{R}} T(R) \cdot \mathbb{S}_{\text{intra}}(R) + \frac{1}{\binom{|T|}{2}} \sum_{R, R' \in \mathcal{R}} T(R) \cdot T(R') \cdot \mathbb{S}_{\text{inter}}(R, R') \quad (3)$$

where $|T| = \sum_{R \in \mathcal{R}} T(R)$.

Thus, the synergy of a multi-robot team is the sum of the average intra-robot synergy of each robot and the average inter-robot synergy of every pair of robots.

The intra and inter-robot synergy equations are modified from the pairwise synergy function of the Synergy Graph model [5], using the intra and inter-robot weights in the SGraCR model. The synergy function is also adapted from the Synergy Graph model, but takes into account the new definitions of intra-robot and inter-robot synergy.

5. LEARNING THE SGRACR FROM DATA

In the previous section, we formally defined the SGraCR model and the synergy equations. To be able to use the SGraCR model on an actual multi-robot problem, we need to learn a SGraCR from data. In this section, we contribute our learning algorithm, that learns a SGraCR using only data of the performance of multi-robot teams.

A team $T : \mathcal{R} \to \mathbb{Z}_0^+$ comprises multiple robots $R \in \mathcal{R}$, and $V(T)$ is an observation of the team T's value at the task. We are given a set of training data D_{train} that contains tuples of teams and their observed values, i.e., $(T, V(T)) \in D_{\text{train}}$. The goal is to find the SGraCR that fits the training data with the highest log-likelihood. The amount of training data we use to learn a SGraCR is much less than that of a Synergy Graph. The Synergy Graph learning algorithm requires multiple observations of all pairs and triples of agents, i.e., $O(n^3)$ where n is the number of agents. In contrast, the SGraCR learning algorithm only requires a single observation per team, and less than $O(|\mathcal{T}|)$ observations, where $|\mathcal{T}|$ is the number of possible teams.

Algorithm 1 shows our learning algorithm. The space of all possible SGraCR structures are exponential in the number of modules, and so it is intractable to completely explore the space. We use simulated annealing to iterate through possible SGraCR structures, and learn the capabilities of the modules using the training data. Simulated annealing serves as an approximation technique for us, and we believe other approximation techniques will perform similarly.

Algorithm 1 Learn SGraCR from training data

LearnSGraCR($\mathcal{M}, D_{\text{train}}$)

```
1: G ← RandomSGraCRStructure(M)
2: C ← EstimateCapabilities(G, D_train)
3: S ← {G, C}
4: l ← LogLikelihood(S, D_train)
5: for k = 1 to k_max do
6:    G' ← NeighborSGraCRStructure(G)
7:    C' ← EstimateCapabilities(G', D_train)
8:    S' ← {G', C'}
9:    l' ← LogLikelihood(S', D_train)
10:   if P(l, l', Temp(k, k_max)) > random() then
11:      S ← S'
12:      l ← l'
13: return S
```

The structure of Algorithm 1 is similar to the Synergy Graph learning algorithm [5]; the key difference lies in these three functions used to learn the SGraCR. The function `RandomSGraCRStructure` that generates a random SGraCR graph structure; `NeighborSGraCRStructure` that generates a neighbor SGraCR structure based on the current structure; and `EstimateCapabilities` that estimates the modules' capabilities using the training data and a SGraCR structure.

The first function, `RandomSGraCRStructure`, generates a random SGraCR graph structure based on the set of modules $\mathcal{M}$, by creating $|\mathcal{M}|$ vertices, and randomly adding edges between pairs of vertices such that the overall graph is connected. The weights of the edges (both intra-robot and inter-robot) are randomly generated to be an integer in the range $[w_{\min}, w_{\max}]$. Having a fixed range of edge weights allows the learning algorithm to effectively explore the space of possible SGraCR structures.

To generate neighbor SGraCR structures, the function `NeighborSGraCRStructure` takes the current structure and modifies it with one of four actions:

1. add a random edge between two vertices;
2. remove a random non-looping edge that does not disconnect the graph;
3. increase one weight of an existing edge;
4. decrease one weight of an existing edge.

Fig. 3 shows the four actions on an example SGraCR graph structure. The four actions change the shortest distance between modules, and hence affect the synergy equations, while ensuring that the SGraCR graph remains connected. The edge weights that are generated or changed respect the integer range $[w_{\min}, w_{\max}]$. Through the four actions, the space of possible SGraCR graph structures is explored.

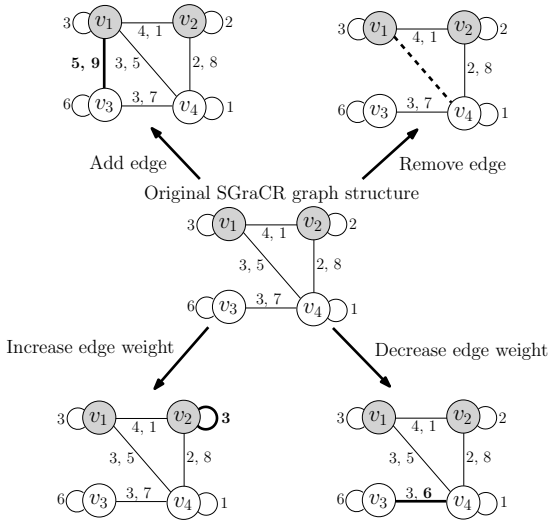


Figure 3: An example SGraCR graph structure, and the four actions used to generate neighbor SGraCR structures.

The first two functions only generate the SGraCR graph structure and not the module capabilities. The last function, `EstimateCapabilities`, estimates the capabilities of the modules using the SGraCR structure and the training data. With the existing SGraCR structure, the synergy formula $\mathbb{S}$ (Eqn. 3) is used to form an equation, where only the Normally-distributed capability variables are unknown. For example, suppose the training data contains $(T, V(T))$, where T consists of a single robot $R = (m_1, m_2)$. Then, $\mathbb{S}(T) = \mathbb{S}_{\text{intra}}(R) = \phi(d_{\text{intra}}(v_{m_1}, v_{m_2}))(C_{m_1} + C_{m_2})$. Since the SGraCR structure is known, d_{intra} is computed and fed into the compatibility function ϕ (that is also known). Thus, the only unknowns in $\mathbb{S}(T)$ are C_{m_1} and C_{m_2}. Since C_{m_1} and C_{m_2} are independent Normally-distributed variables, the log-likelihood of $V(T)$ given $\mathbb{S}(T)$ can be written out with unknowns $\mu_1, \mu_2, \sigma_1^2, \sigma_2^2$ that correspond to the means and variances of C_{m_1} and C_{m_2} respectively.

Each training data is converted into a log-likelihood expression where the means and variances of module capabilities are unknowns. The capabilities are then found by using a non-linear solver to maximize the log-likelihood expressions. Overall, SGraCR structures are altered and capabilities learned, in order to create a candidate SGraCR that is compared to the current best-guess. Through simulated annealing, the algorithm converges on an approximation of the optimal SGraCR that maximizes the log-likelihood.

6. FORMING THE MULTI-ROBOT TEAM

We have formally defined the SGraCR model and contributed our learning algorithm. In this section, we introduce our team formation algorithm that finds the δ-optimal team. The δ-optimal team T^*_δ must be feasible ($F(T^*_\delta) = 1$), within the cost threshold ($\text{cost}_\mathcal{T}(T^*_\delta) \leq c_{\max}$), and attain a value of at least v with probability δ.

The feasibility function F, cost function, cost threshold $c_{\max}$ and probability δ are domain-specific and given, but the value v is not. Further, the synergy function defined in Eqn. 3 returns a Normally-distributed variable. Thus, we need to convert a Normally-distributed variable into a single value. To do so, we use the V function of the Synergy Graph model [5], that we rename to `Evaluate` in this paper, that takes as input a random Normally-distributed variable $X \sim \mathcal{N}(\mu_X, \sigma^2_X)$ and a risk-factor ρ:

$$\texttt{Evaluate}(X, \rho) = \mu_X + \sigma_X \cdot \Phi^{-1}(\rho) \quad (4)$$

In particular, $\texttt{Evaluate}(X, 1-\delta)$ returns a value v such that $P(X \geq v) = \delta$. Further, with two variables X_1 and X_2, and $\texttt{Evaluate}(X_1, 1-\delta) = v_1$, $\texttt{Evaluate}(X_2, 1-\delta) = v_2$, if $v_1 \geq v_2$ then $P(X_2 \geq v_1) \leq \delta$.

Thus, finding the δ-optimal team is equivalent to finding the team T^* that maximizes $\texttt{Evaluate}(\mathbb{S}(T^*), 1-\delta)$. To do so, we use simulated annealing to explore the space of possible multi-robot teams. Algorithm 2 shows the team formation algorithm. The inputs to the algorithm are: S, the SGraCR model; F, the feasibility function; cost, the cost function; and $c_{\max}$, the maximum cost of the team.

Algorithm 2 Find the δ-optimal team

FindOptimalTeam($S, F, \text{cost}, c_{\max}, \delta$)

```
1: T* ← GenerateRandomTeam(S, cost, c_max)
2: v* ← Evaluate(S(T*), 1 − δ)
3: for k = 1 to k_max do
4:    T ← RandomNeighbor(T*, S, cost, c_max)
5:    v ← Evaluate(S(T), 1 − δ)
6:    if P(v*, v, Temp(k, k_max)) > random() then
7:       T* ← T
8:       v* ← v
9: return T*
```

`RandomNeighbor` generates neighbor candidates from the existing team with three possible actions:

1. a random robot R_{existing} is removed;
2. a random robot R_{new} is created;
3. a module on an existing robot R_{existing} is changed.

Actions 1 and 2 (removing and creating robots) involve changing the function $T^* : \mathcal{R} \to \mathbb{Z}^+_0$ slightly. Action 1 picks a random robot $R_{\text{existing}} \in \mathcal{R}$ such that $T^*(R_{\text{existing}}) > 0$ and returns the team $T(R) = \begin{cases} T^*(R) - 1 & \text{if } R = R_{\text{existing}} \\ T^*(R) & \text{otherwise} \end{cases}$. Similarly, Action 2 randomly selects a robot $R_{\text{new}} \in \mathcal{R}$ and increases the number of that robot in the team by 1, i.e., $T(R) = \begin{cases} T^*(R) + 1 & \text{if } R = R_{\text{new}} \\ T^*(R) & \text{otherwise} \end{cases}$.

Action 3 first picks a random robot $R_{\text{existing}} \in \mathcal{R}$ such that $T^*(R_{\text{existing}}) > 0$. Suppose $R_{\text{existing}} = (m_1, \ldots, m_N)$. Action 3 then picks a random number $n \in [1, N]$ and changes module M_n in the robot to be $m'_n \neq m_n$. Hence, the new robot $R_{\text{new}} = (m_1, \ldots, m_{n-1}, m'_n, m_{n+1}, \ldots, m_N)$; R_{new} differs from R_{existing} by only one module.

These 3 actions generate candidate teams that effectively explore the space of all teams, but the teams may not be feasible and/or be over the cost threshold. Thus, if $F(T) = 0$ or $\text{cost}(T) > c_{\max}$, the actions are repeated until a suitable team is generated. The difficulty of generating a feasible team within the cost threshold is domain-dependent since it depends on F, cost, and $c_{\max}$. Once a neighbor team is formed, its synergy is computed and converted into a real number by `Evaluate`. The new team's value is then compared to the existing and accepted subject to the temperature schedule of the simulated annealing algorithm.

Thus, after all $k_{\max}$ iterations of simulated annealing is complete, the best team found, T^*, is returned, and is approximately equal to the δ-optimal team T^*_δ.

7. EXPERIMENTS AND RESULTS

In the previous sections, we introduced the Synergy Graph for Configurable Robots (SGraCR) model, and contributed algorithms for learning the model and forming the team. In this section, we describe extensive experiments that demonstrate the efficacy of our model and algorithms. First, we show that our algorithms perform well with synthetic data derived from a hidden SGraCR model. Next, we demonstrate the efficacy of our SGraCR model and algorithms on a simulated manufacturing scenario. Finally, we apply our model and algorithms on a real robot scenario, and show that it is capable of selecting modules to form an effective multi-robot team. While we use a manufacturing scenario in our experiments, the SGraCR model and algorithms are applicable to a wide range of multi-robot domains.

To evaluate the SGraCR model and algorithms, we compared our performance to two benchmarks. First, we used the Synergy Graph model to learn from the training data and form a multi-robot team [5]. Since the Synergy Graph models each robot separately, the size of the Synergy Graph was much larger than that of the SGraCR. Also, we evaluated the performance of the IQ-ASyMTRe algorithm [9]. The ASyMTRe algorithm is well-known to form multi-robot teams in tightly-coupled scenarios, where robots are modeled as collections of schemas (similar to our modules, although we do not consider the inputs and outputs). We used a least-squares solver that analyzed the training data and assigned costs to each module, and used their cost function to rank possible multi-robot teams.

For each set of experiments, we scored the teams found by SGraCR, Synergy Graph, and IQ-ASyMTRe as the number of standard deviations away from the mean of all possible teams. So, if the mean and variance of the values of all possible teams are μ and σ^2 respectively, and the team found had a value v, then the score was $\frac{v-\mu}{\sigma}$.

7.1 Experimental Setup

In all three sets of experiments, we used the same problem domain — manufacturing. The task involved transporting some items from location L_0 to perform drilling (at location L_1) and then milling (at location L_2) and finally delivered to a destination location L_3. The manufacturing floor had pre-existing drilling and milling stations at fixed locations (L_1 and L_2 respectively), and the goal was to form a multi-robot team that would move all the items through the manufacturing plan. All the robots would be mobile, and

had varying behaviors. Robots could also be configured to perform drilling *or* milling, which would allow the items to be transported past locations. For example, if a robot could drill, then it could transport items from L_0 directly to L_2 for milling, bypassing L_1 since the robot performs the drilling.

We defined 3 types of modules: M_1, M_2, M_3. $M_1 = \{\text{slow, medium, fast}\}$ are the motors, $M_2 = \{1, 2, 3, 4\}$ are the carrying capacities, and $M_3 = \{B_{0,1}, B_{0,2}, B_{1,2}, B_{1,3}, B_{2,3}\}$ encapsulate both the software programmed into the robots and drilling/milling capabilities. A behavior $B_{i,i+1}$ implies that the robot only transports items from L_i to L_{i+1}. A behavior $B_{i,i+2}$ implies that the robot also performs drilling/milling, e.g., $B_{1,3}$ means that a robot transports drilled items from L_1 to location L_3 and performs milling on the item.

The feasibility function F returns 1 iff the team of robots are able to drill, mill and transport all items to L_3. For all our experiments below, we set $\delta = 0.5$, so the goal was to find the team that attains the highest mean value. Faster motor speeds, higher carrying capacities, and adding drilling/milling functionality had higher costs. The cost threshold $c_{\max}$ was set such that the maximum number of robots was five for the synthetic and simulation experiments, and three for the real robot experiments.

In each trial, we generated a set of training data. The learning algorithm uses the training data to learn a SGraCR model, and the team formation algorithm uses the learned SGraCR to find the team that approximates the δ-optimal team. A similar approach was used to learn a Synergy Graph and form a team, and for IQ-ASyMTRe the training data set was used to estimate the module costs.

7.2 Synthetic Data

In our first set of experiments, we used synthetic data derived from a hidden SGraCR model. Using the experimental domain described above, we generated a hidden SGraCR model with 12 vertices and randomly generated the module capabilities. The hidden model was used to create 100 training data $(T, V(T)) \in D_{\text{train}}$, where the value $V(T) = \texttt{Evaluate}(\mathbb{S}(T), 1-\delta)$ of the hidden model. The training data was used to learn a new SGraCR model. Finally, our team formation algorithm was run on the learned SGraCR model, and its value calculated using the hidden model, i.e., if the algorithm selected team T, then $V(T) = \texttt{Evaluate}(\mathbb{S}(T), 1-\delta)$ of the *hidden* model. This process is very similar to the experiments on the Synergy Graph model [5], and we wanted to compare our new SGraCR model with the Synergy Graph model using this benchmark.

We performed 20 trials, where a different hidden SGraCR model was generated each time. The second column of Table 1 shows the results of our trials with synthetic data. SGraCR vastly outperformed the Synergy Graph model and IQ-ASyMTRe. We believe that this is largely because the data was derived from a hidden SGraCR model. The low performance of the Synergy Graph compared to SGraCR shows that SGraCR is a more expressive model; otherwise, the Synergy Graph would have a similar score to SCraCR.

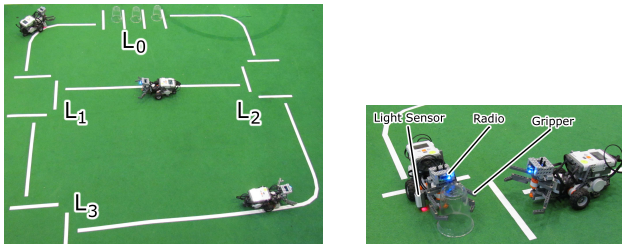
7.3 Simulated Robots

In our second set of experiments, we created a 2D simulator, where mobile robots moved to transport items from one location to another. The value of a team was the negative of the number of timesteps taken to transport 100 items from L_0 to L_3, i.e., if a team T took x timesteps then $V(T) = -x$.

We ran the simulator on all 6056 possible teams to calculate their value, and ran 20 trials. In each trial, 100 data points of the 6056 was used for training, so only a small subset of possible teams was visible by the learning algorithm to learn a SGraCR. The team formation algorithm then searched the learned SGraCR to approximate the δ-optimal team. The third column of Table 1 shows the results of the simulated experiments. SGraCR and Synergy Graph both perform very well, finding teams with scores of 1.33, which indicates that the simulated domain can be sufficiently modeled with the Synergy Graph. However, although the Synergy Graph model has a similar performance, it contains 60 vertices compared to SGraCR's 12, showing that the Synergy Graph model does not scale as well as the SGraCR model to more complex scenarios involving modular robots. Thus, the SGraCR model is well-suited for configurable robots in multi-robot teams.

7.4 Real Robot Experiments

In our final set of experiments, we used Lego NXT robots in a pseudo-manufacturing setting. We chose the Lego platform as the hardware is modular and configurable to fit any task. We designed the robot task such that it involved manipulation and movement, which are essential components of many robot domains. Since we only used the time of task completion to train the SGraCR model, the approach in our experiments would be identical if any other robot platform or task was used. Fig. 4a shows the layout of our robot experiments, and Fig. 4b shows a NXT robot approaching L_1, with some of its components labeled. Each robot was programmed to follow a white line from station to station, and pass transparent cups to each other. The drilling/milling operations were not actually performed but assumed to take place either at the stations or by the robot transporting it.



(a) (b)

Figure 4: a) The layout of the real robot experiments involving NXT robots transporting items from L_0 to L_3. b) A NXT robot as it approaches L_1.

Due to the limited carrying capacity of the NXT robots, we set the carrying capacity modules $M_2 = \{1\}$. Also, we had the physical limitation of 3 NXT robots, so teams had a maximum size of 3. As such, $|\mathcal{T}| = 45$. In each trial, the robots moved 3 items from the start location L_0 to the end L_3, handing 1 item to each other at each station. The value of a team was the negative of the cost and the time taken, i.e., a team T with cost c and took x seconds to transport all 3 items had a score of $V(T) = -c - x$. We reduced the value of a team by its cost for two reasons: to show the efficacy of the SGraCR model over different value functions (compared to the previous subsections), and to better reflect that the cost of a team has an effect in a manufacturing scenario.

Approach	Score		
	Synthetic Data	Simulation	Real Robots
SGraCR	1.77 ± 1.64	1.33 ± 0.52	0.86 ± 0.46
Synergy Graph	0.56 ± 1.45	1.33 ± 0.29	0.37 ± 0.82
IQ-ASyMTRe	0.93 ± 1.99	0.41 ± 0.54	0.59 ± 0.23

Table 1: Experimental results of SGraCR and two competing approaches using synthetic data derived from a hidden SGraCR model, simulated robots in a manufacturing scenario, and robot experiments using Lego NXT robots.

Similar to the above sets of experiments, we used a subset of the data collected to learn a SGraCR. We then formed a multi-robot team using the learned SGraCR. In these experiments, $|D_{\text{train}}| = 20$, which is less than half of $|\mathcal{T}|$. The SGraCR and Synergy Graph models have 9 and 15 vertices respectively, and so we chose a training size of 20 to provide enough information to solve for the unknowns. We ran 100 trials, where each trial used a different subset of 20 training data. The last column of Table 1 shows the results of our robot experiments. SGraCR outperforms the Synergy Graph and IQ-ASyMTRe approaches, demonstrating that SGraCR captures interactions that are unmodeled by the Synergy Graph model and IQ-ASyMTRe. We performed a one-tailed paired T-test on the results, and found that SGraCR has a statistical significance of $p = 3.7 \times 10^{-7}$ against the Synergy Graph model, and a statistical significance of $p = 8.0 \times 10^{-9}$ against IQ-ASyMTRe. Thus, the SGraCR model is robust and well-equipped to be applied to robot domains involving configurable multi-robot teams.

8. CONCLUSIONS

We formally introduced the Synergy Graph for Configurable Robots (SGraCR) model, where each robot module is a vertex in a weighted connected graph, and distances between vertices in the graph are inversely related to how well modules work together. Edges have intra and inter-robot weights, to distinguish between the distance of modules within a single robot and modules across robots. We defined intra and inter-robot synergy, and the synergy of a multi-robot team composed of modules. We assume no prior knowledge of the capabilities of the robots and modules, and contribute a learning algorithm that learns a SGraCR using a small set of training data. We also defined the notion of a δ-optimal team, and contributed a team formation algorithm that approximates it, using the learned SGraCR.

We performed extensive experiments in simulation and on real robots to demonstrate the efficacy of our model and algorithms. First, we showed that the learning and team formation algorithms perform well when using data derived from a hidden SGraCR. Next, we used a simulated manufacturing scenario and showed that our model is applicable to this domain. The multi-robot team formed by module selection outperforms the IQ-ASyMTre algorithm and performs similarly to the Synergy Graph model, although SGraCR uses less vertices and scales better as the number of modules increases. Lastly, we used real robots in a manufacturing scenario, and showed that SGraCR forms an effective multi-robot team that outperforms the other approaches.

While our experiments used a manufacturing scenario, the SGraCR model is applicable to many general robot domains. As robots are increasingly modular and reconfigurable, many multi-robot problems require the composing of teams by selecting modules to best fit the task. In addition, since SGraCR models each module and its capabilities, the capability of robots (combinations of modules) that have not been seen during training can be inferred from the model, and the performance of novel multi-robot team combinations deduced. Our learning algorithm does not require prior knowledge of the capabilities of the modules, so our approach is also applicable to ad hoc domains, where robots and/or modules have not worked together.

Acknowledgments

The authors thank Marcos Maximo for his work with the NXTs. This work was partially supported by the Air Force Research Laboratory under grant no. FA87501020165, by the Office of Naval Research under grant number N00014-09-1-1031, and the Agency for Science, Technology, and Research (A*STAR), Singapore. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

9. REFERENCES

[1] B. Banerjee and L. Kraemer. Coalition Structure Generation in Multi-Agent Systems with Mixed Externalities. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 175–182, 2010.

[2] S. Barrett, P. Stone, and S. Kraus. Empirical Evaluation of Ad Hoc Teamwork in the Pursuit Domain. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 567–574, 2011.

[3] M. de Weerdt, Y. Zhang, and T. Klos. Distributed Task Allocation in Social Networks. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 500–507, 2007.

[4] M. B. Dias and A. Stentz. Multi-Robot Exploration Controlled By A Market Economy. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2714–2720, 2002.

[5] S. Liemhetcharat and M. Veloso. Modeling and Learning Synergy for Team Formation with Heterogeneous Agents. In *Proc. Int. Conf. Autonomous Agents Multiagent Systems*, pages 365–374, 2012.

[6] T. Rahwan, T. Michalak, N. Jennings, M. Wooldridge, and P. McBurney. Coalition Formation with Spatial and Temporal Constraints. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 257–263, 2009.

[7] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition Structure Generation with Worst Case Guarantees. *Journal of Artificial Intelligence*, 111:209–238, 1999.

[8] T. Service and J. Adams. Constant factor approximation algorithms for coalition structure generation. *Journal of Autonomous Agents and Multi-Agent Systems*, 23:1–17, 2011.

[9] Y. Zhang and L. Parker. Task allocation with executable coalitions in multirobot tasks. In *Proc. IEEE Int. Conf. Robotics Automation*, 2012.